WWW.SCIAM-MAGAZINE.NAROD.RU

Занимательный компьютер

О конечном автомате и простейших моделях мышеловок, рибосом и человеческой души

БРАЙАН ХЭЙЕС

У САМЫХ мощных вычислительных машин, которые пока существуют лишь в нашем воображении, нет ни аппаратуры, ни программного обеспечения. Известнейшая среди этих абстрактных машин была придумана английским математиком Тьюрингом в 1936 г. Круг ее возможностей шире, чем у любого действующего компьютера на кремниевых кристаллах, и разумеется, она может вычислить все, что вообще поддается вычислению. Всем прочим умозрительным вычислительным машинам недостает всемогущества машины Тьюринга, но они не менее интересны. Их называют машинами с конечным числом состояний или конечными автоматами, и по их «возможностям» устанавливают минимальные технические требования к цифровой вычислительной машине.

Точное определение конечного автомата потребовало бы строгого математического изложения. Мы же в данном случае ограничиваемся рассмотрением нескольких конкретных примеров. Отправившись на поиски конечных автоматов, я нашел отличный образчик на Лексингтон-авеню в нью-йоркском метро. Это был турникет старого образца (а не распространенная ныне компактная стальная вертушка) в виде дубовой крестовины, до блеска отполированной людскими руками.

Турникет имеет два состояния: заблокирован и разблокирован. Предположим, что он заблокирован, так что крестовину нельзя повернуть. Опускание жетона в прорезь каким-то образом изменяет внутренний механизм, что позволяет турникету поворачиваться; другими словами, жетон вызывает переход в состояние «разблокирован». Поворот крестовины на 90° приводит к другому переходу, что возвращает турникет в состояние «заблокирован». Переходы схематически показаны на верхнем рисунке на с. 99. Состояния системы представлены узлами (прямоугольниками), а переходы — дугами (стрелками) между ними.

НА ЯЗЫКЕ конечных автоматов опускание жетона и нажатие на поручень турникета — это возможные входы системы. Реакция автомата зависит и от входа, и от состояния в момент входа. Нажатие на поручень до того, как турникет получит жетон, не поможет вам попасть на станцию. Опускать жетон, когда турникет уже разблокирован, также бессмысленно, хотя по несколько иным причинам. Второй жетон будет принят, но это не отразится на состоянии автомата: пропускается один человек, а затем турникет снова блокируется. Три или четыре жетона подряд также будут приняты, но турникет пропустит только одного человека. Скептики потребовали бы больше данных, прежде чем сделать обобщение: все жетоны, опущенные после первого, не оказывают никакого действия. Но пусть они экспериментируют со своими жетонами.

Турникет не может пропустить нескольких человек за несколько опущенных подряд жетонов по той причине, что у него нет средства для подсчета полученных им жетонов. Он обладает единственным, довольно примитивным видом памяти: при изменении состояния «запоминается», был ли самый последний вход опусканием жетона или нажатием на поручень. Все более ранние входы теряются. Следует заметить, что от этой забывчивости городскому хозяйству нет никакого ущерба. Могло быть хуже: турникет можно было бы спроектировать так, чтобы он менял состояние после каждого опущенного жетона независимо от того, в каком состоянии он находится в данный момент; в этом случае два жетона подряд привели бы турникет в состояние «заблокирован».

Этот пример иллюстрирует большинство основных свойств конечного автомата. Автомат, очевидно, должен иметь некоторое число состояний, причем обязательно конечное. Должны быть входы и выходы, соответствующие любому состоянию. Состояния должны быть дискретными, или легко различимыми, а переходы между ними — действительно мгновенными. Здесь, разумеется, многое зависит от точки зрения: день и ночь — это дискретные состояния, если определить восход и заход как мгновенные процессы. Набор состояний, входы и выходы — вот и весь автомат; он может работать без вспомогательных устройств, и в частности без каких-либо средств хранения информации.

Правила построения конечного автомата допускают некоторые варианты. Существуют детерминированные и недетерминированные автоматы, автоматы Мура и автоматы Мили. В детерминированном автомате данный вход в данном состоянии неизменно приводит к одному и тому же результату; в недетерминированном автомате может быть несколько возможных переходов. В автомате Мура (по имени Э. Мура) каждому состоянию соответствует единственный выход. В автомате Мили (по имени Дж. Мили) выходы соответствуют переходам, а не состояниям. Оказывается, однако, что это многообразие структур — всего лишь иллюзия. Любая задача, выполнимая конечными автоматами одного рода, может быть также выполнена автоматами другого рода, хотя число необходимых состояний может меняться. Здесь я буду преимущественно рассматривать детерминированные автоматы Мура, которые обладают простейшей структурой.

СТОИТ пуститься на поиски конечных автоматов, как вы начнете находить их повсюду. Устройства, приводимые в действие монетой, — излюбленные примеры из учебников. Некоторые торговые автоматы не столь ненасытны, как турникет в метро: как только они получают нужное количество денег, они переходят в состояние, при котором все лишние монеты возвращаются. Устройство, приводимое в действие монетой и имеющее самое большое число состояний, — это, без сомнения, типичный лас-вегасский игорный автомат. В принципе он детерминирован, тем не менее при отыскании такого входа (монета и нажатие на рукоятку), который бы вызвал переход к желаемому заключительному состоянию, успех весьма сомнителен.

Многие бытовые приборы можно рассматривать как конечные автоматы, хотя и довольно «тупые». Стиральная машина проходит неизменную последовательность состояний — залив, стирка, полоскание, отжим, — и лишь несколько осмысленных входов, таких, как отключение машины от сети, приводит к одному и тому же результату при любом состоянии. Точно так же светофор имеет небольшой диапазон состояний, которые обычно повторяются. По-моему, самый скучный из всех конечных автоматов — электронные часы. Если они показывают месяц, число, часы, минуты и секунды, то имеют около 31 млн. состояний. В течение года они находятся в каждом состоянии ровно один раз.

Мышеловка — тоже конечный автомат: мышь, к своему несчастью, вызы-

WWW.SCIAM-MAGAZINE.NAROD.RU

вает переход от состояния с поднятой дверцей к состоянию с захлопнутой дверцей. Замок с шифром — конечный автомат со многими возможными входами, из которых только один приводит к переходу в другое состояние. Телефон имеет состояния, которые можно обозначить так: трубка на рычаге, трубка снята с рычага, сигнал «занято», сигнал «свободно», набор номера, звонок, соединение и неисправен. На примере автомобиля можно увидеть, что результат того или иного входа изменяется в зависимости от текущего состояния системы. Что происходит, если вы нажимаете на педаль акселератора? Это зависит от того, едет ли машина, включено ли сцепление, снята ли машина с ручного тормоза, включена ли передача — и на какой ход, передний или задний, — а также от того, открыта ли дверь гаража.

В живой клетке молекулярная система, состоящая из рибосомы и различных видов тРНК, действует как конечный автомат. Входами являются четыре нуклеотида мРНК, обозначенные сокращениями U, A, G и C. Выходами служат 20 аминокислот, из которых состоят белки. Цепочка нуклеотидов распознается как правильный вход для этого автомата, если только она состоит из начальной последовательности AUG. Затем автомат непрерывно считывает входную последовательность, каждый раз меняя состояние, когда распознается кодон, или триплет нуклеотидов. Три особо выделенных кодона UAA, UAG и UGA являются сигналами останова: когда встречается один из них, автомат останавливается. Весьма полезно представление и многих других биологических систем в виде конечных автоматов; первым, пожалуй, приходит на ум пример гемоглобина и бактериальных белков, являющихся активаторами или репрессорами.

Даже „душа" в теологии Фомы Аквинского не что иное, как тщательно разработанный и полностью детерминированный конечный автомат. При сотворении она находится в состоянии осуждения за первородный грех. При крещении она переходит в состояние благодати, однако некоторые поступки (идолопоклонство, богохульство, прелюбодеяние и т.п.) вызывают переход в состояние греха. Тогда для возвращения души в состояние благодати необходимы исповедь, раскаяние и отпущение грехов. Результат самого последнего входа, смерти, очень сильно зависит от состояния души в этот момент: в состоянии благодати смерть приводит к спасению, а в состоянии греха — к осуждению на вечные муки. Механизм души, разумеется, гораздо сложнее, чем даваемое этим описанием представление о нем. При полном описании надо было бы различать степени греха (простительный и смертный, случайный и привычный) и ввести другие возможные состояния души (те, которые связаны с преддверием ада и чистилищем), а также другие возможные входы (например, Страшный суд).

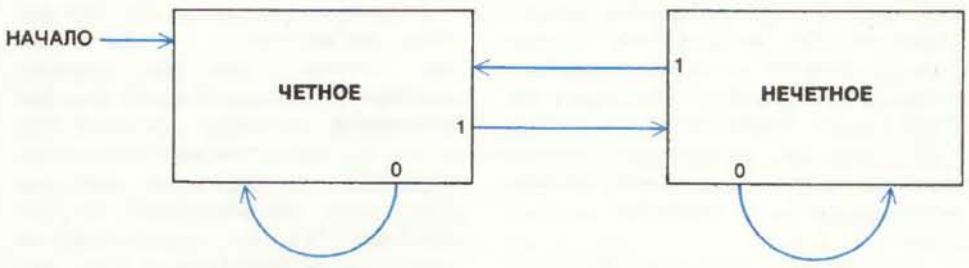
В квантовой механике конечным автоматом становится даже атом и, следовательно, все, что состоит из атомов. Состояния атомов — это разрешенные уровни энергии, входами и выходами служат фотоны, или кванты электромагнитного излучения. Я думаю, что при точном описании атом был бы классифицирован как недетерминированный автомат Мили с эпсилон-переходами. Он недетерминированный, потому что результат входа нельзя точно предсказать. Это автомат Мили, потому что природа выхода (а именно энергия фотона) определена переходом, а не состоянием, в которое он перешел. (Эпсилон-переходами называются такие переходы, которые могут произойти в отсутствие какого бы то ни было входа; их следует включить в модель, поскольку атом может испустить фотон и изменить свое состояние спонтанно.)

ЯВЛЯЕТСЯ ли мозг человека конечным автоматом? Дело в том, что современное изучение систем с конечным числом состояний началось с модели нейронной сети, введенной в 1943 г. У. Мак-Каллоком и У. Питтсом. Нейроны Мак-Каллока и Питтса представляли собой простые клетки со входами «возбуждение» и «торможение»; у каждой клетки был единственный выход и два внутренних состояния: возбуждения и покоя. Клетки могли образовывать цепи, реализующие различные логические функции: в том числе «и», «или» и «нет», которые теперь являются самыми обычными функциями логической схемы в электронных системах. Эквивалентность идеализированной нейронной сети с диаграммой переходов того типа, который показан здесь, была установлена в 1956 г. С. Клини из Висконсинского университета в Мадисоне.

И через сорок лет после появления работы Мак-Каллока и Питтса все еще обсуждается вопрос, можно ли с достаточным основанием квалифицировать мозг как систему с конечным числом состояний. Разумеется, число нейронов обязательно конечно, но дело не только в этом. Реальный нейрон гораздо сложнее клетки с двумя состояниями, и некоторые его свойства могут меняться непрерывно, а не ограничиваться дискретными состояниями. Кроме того, отсутствие возможности хранить дополнительную информацию для конечного автомата по меньшей мере неудобно. Если духовная жизнь не более чем последовательность мгновенных состояний без знания своей собственной истории, то что же такое память?

Эмоциональные состояния, являющиеся предметом обсуждения в психологии, такие, как скука, страх, тоска, экстаз и горе, по-видимому, более подвластны теории конечных автоматов. В то же время состояний так много, а переходы еще настолько не изучены и непонятны, что такая модель бесполезна. Только для низших животных возможно наметить нечто большее, нежели изолированные фрагменты диаграммы переходов, а человек в экспериментах с этими видами не получил прямого доступа к предполагаемым состояниям психики. Действительно, боль-

Диаграмма переходов для турникета метро



Автомат для проверки на четность

WWW.SCIAM-MAGAZINE.NAROD.RU

шая работа в этом направлении была проделана бихевиористами, которые отрицают само существование состояний психики.

Случай с цифровой вычислительной машиной (а здесь я имею в виду материальную машину, аппаратуру) также проблематичен. Распространенная умозрительная модель вычислительной машины, построенная Дж. фон Нейманом, делит машину на центральный процессор и массив ячеек памяти. Нет сомнения, что многие компоненты центрального процессора, такие, как регистры, сумматоры и механизмы управления, которые руководят внутренними операциями процессора, можно описать в терминах конечных автоматов.

Неприятности начинаются, когда дело доходит до памяти. По правилам построения конечного автомата внешняя память не допускается, и, таким образом, каждую ячейку надо рассматривать не как запоминающее устройство, отдельное от процессора, а как часть состояния всего автомата. Если все ячейки пусты, то компьютер находится в одном состоянии, если хотя бы одна заполнена, автомат перешел в другое состояние и т.д. Подобная концепция компьютера особенно бессмысленна, в частности, потому, что она не устанавливает связи между состоянием машины и тем, что она делает. Кроме того, число состояний огромно. Примитивный компьютер, способный запомнить информацию в 100 бит, не сможет, по-видимому, пройти через все свои состояния, даже если будет постоянно работать на протяжении существования Вселенной.

ПЕРВОСТЕПЕННАЯ роль конечного автомата в теории вычислительных машин заключается в более высоком уровне абстракции, чем у обычного механизма аппаратуры. Вычислительная машина, работающая под руководством программы, это уже не нагромождение логических схем, регистров, ячеек памяти и других атрибутов электроники: это «виртуальная» машина, рабочие части которой определены программой и, если надо, могут быть определены заново. В то время как аппаратура знает лишь двоичные числа и простые команды для их перемещения и действий над ними, виртуальная вычислительная машина имеет дело с более выразительной символьной системой: словами, уравнениями, массивами, функциями, векторами, кодонами, списками, образами, быть может, даже с идеями. Теория конечных автоматов ценна при создании виртуальных машин, и иногда виртуальная машина сама оказывается конечным автоматом.

Рассмотрим программу, цель которой — считывание последовательности двоичных разрядов (единиц и нулей) и выдача сообщения о том, четным или нечетным получилось число единиц. (Эта работа не лишена практической пользы: например, такие программы проверки на четность используются для обнаружения ошибок при передаче двоичных данных по телефону.) Программа может быть построена как конечный автомат с двумя состояниями, как показано на нижнем рисунке на с. 99. Операция начинается в четном состоянии, так как первоначально не получено ни одной единицы, а нуль считается четным числом. Каждая единица во входной последовательности приводит к изменению состояния, в то время как получение нуля в любом состоянии не изменяет его. Хотя машина не может «запомнить» никакого входа, кроме самого последнего, и, разумеется, не может считать единицы и нули, ее выход всегда отражает четность входной последовательности.

Модель вычисления с конечным числом состояний очень часто встречается в программах, так или иначе связанных с текстом или другой информацией, которая принимает лингвистическую форму. Замечательным примером служат компиляторы: программы, транслирующие операторы программы на входном языке в эквивалентные операторы на выходном языке, чаще называемом «машинным языком» данного компьютера. Компиляторы и другие программы — трансляторы — основные понятия, связанные с виртуальными машинами. Они являются посредниками между символами, имеющими смысл для человека, и символами, распознаваемыми вычислительной машиной.

Часть компилятора, которую можно представить как конечный автомат, называется сканером или лексическим анализатором. Подобно турникету метро, он тоже «пожирает жетоны». Но в этом случае, однако, в качестве «жетонов» выступают слова, или основные лексические единицы языка. Лексический анализатор распознает каждую группу символов и определяет, является ли она настоящим «жетоном», таким, как команда или число: если нет, то лексический анализатор отбрасывает ее как бессмыслицу, подобно турникету, не принимающему фальшивый жетон.

Операцию, производимую лексическим анализатором, можно проиллюстрировать с помощью конечного автомата, предназначенного для распознавания «жетонов» простого языка, пусть и с весьма ограниченными выразительными средствами: «жетоны» образованы исключительно из римских цифр. На самом деле допускаются только числа специального вида: они получаются путем сложения, так что 9

Издательство **МИР** *предлагает:*

З. Лойда, Р. Госсрау, Т. Шиблер

ГИСТОХИМИЯ ФЕРМЕНТОВ

Лабораторные методы

Перевод с английского

Предлагаемое вашему вниманию пособие основано на самых последних методических достижениях гистохимии ферментов. Подробно рассмотрены более 100 методов гистохимического выявления ферментов, причем многие методы изложены в собственных удачных модификациях авторов, ведущих специалистов в этой области.

В книге много ценных технических рекомендаций, которые так важны при воспроизведении методик. Кроме того, обсуждаются возможные причины артефактов, столь частых в лабораторной практике. Приведены материалы по подготовке тканей для гистохимического исследования ферментов (отбор и обработка тканей, фиксация, промывка, приготовление срезов, заключение, инкубация и др.). Раздел, посвященный отдельным ферментам, весьма обширен — описываются методы выявления гидролаз, гликозидаз, пептидаз, трансфераз, лиаз, оксидоредуктаз.

Следует подчеркнуть, что пособие имеет практическую направленность, вопросы теории в нем обсуждаются в объеме, который необходим для понимания проводимых реакций и сознательного выбора из ряда аналогичных методов наиболее адекватного целям исследования.

Книга, несомненно, будет с интересом встречена специалистами-гистохимиками, а также гистологами, цитологами, биохимиками. Она пригодится врачам-лаборантам, студентам, аспирантам и преподавателям университетов, медицинских, сельскохозяйственных и педагогических институтов.

1982, 270 с. Цена 1 р. 60 к.

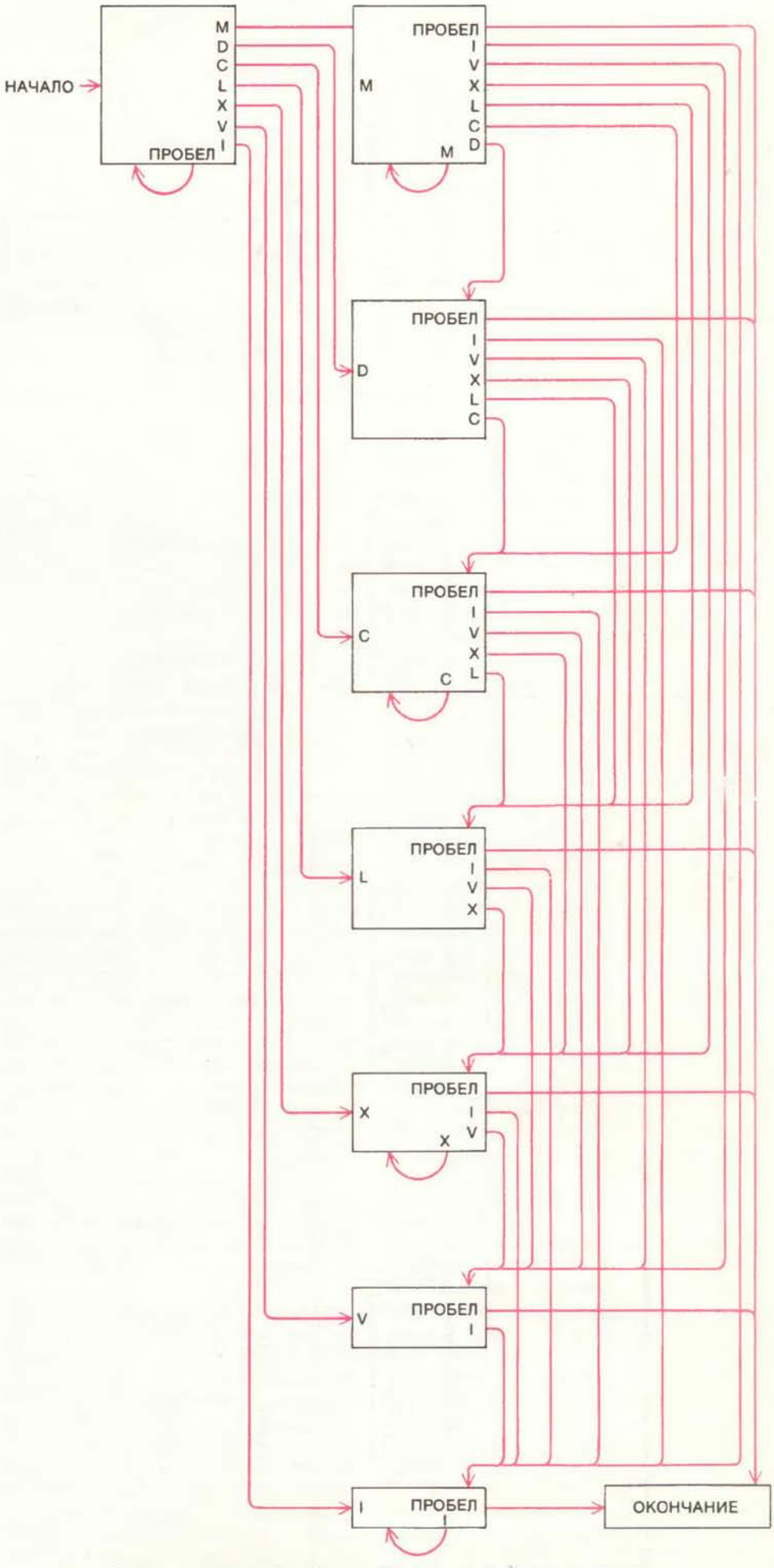
WWW.SCIAM-MAGAZINE.NAROD.RU

записывается как VIIII, а не IX. (Имеются свидетельства, что сами римляне пользовались именно такой записью; представление при помощи вычитания является, по-видимому, более поздним германским новшеством.)

Диаграмма переходов для автомата, обрабатывающего числа, записанные римскими цифрами, показана справа. Ее входной алфавит состоит из литер M, D, C, L, X, V и I, а также символов пробела, или просто пробелов. Любой первоначальный пробел попросту игнорируется, но, как только получена первая литера, программа немедленно переходит в состояние, идентифицируемое (для удобства) названием литеры. Если первая литера есть M, то за ней может следовать любая литера из алфавита, включая другую литеру M. Если следующая литера D, то ситуация несколько меняется. Из состояния D не определен переход обратно в состояние M, потому что любая последовательность символов, которая содержит DM, не может быть правильно составленным «жетоном»: такое число нельзя образовать из римских цифр путем сложения. Кроме того, невозможен переход из состояния D в состояние D, так что DD тоже исключенная последовательность. Причина здесь в том, что для «половинок» D, L и V в римской нумерации повторения не допускаются.

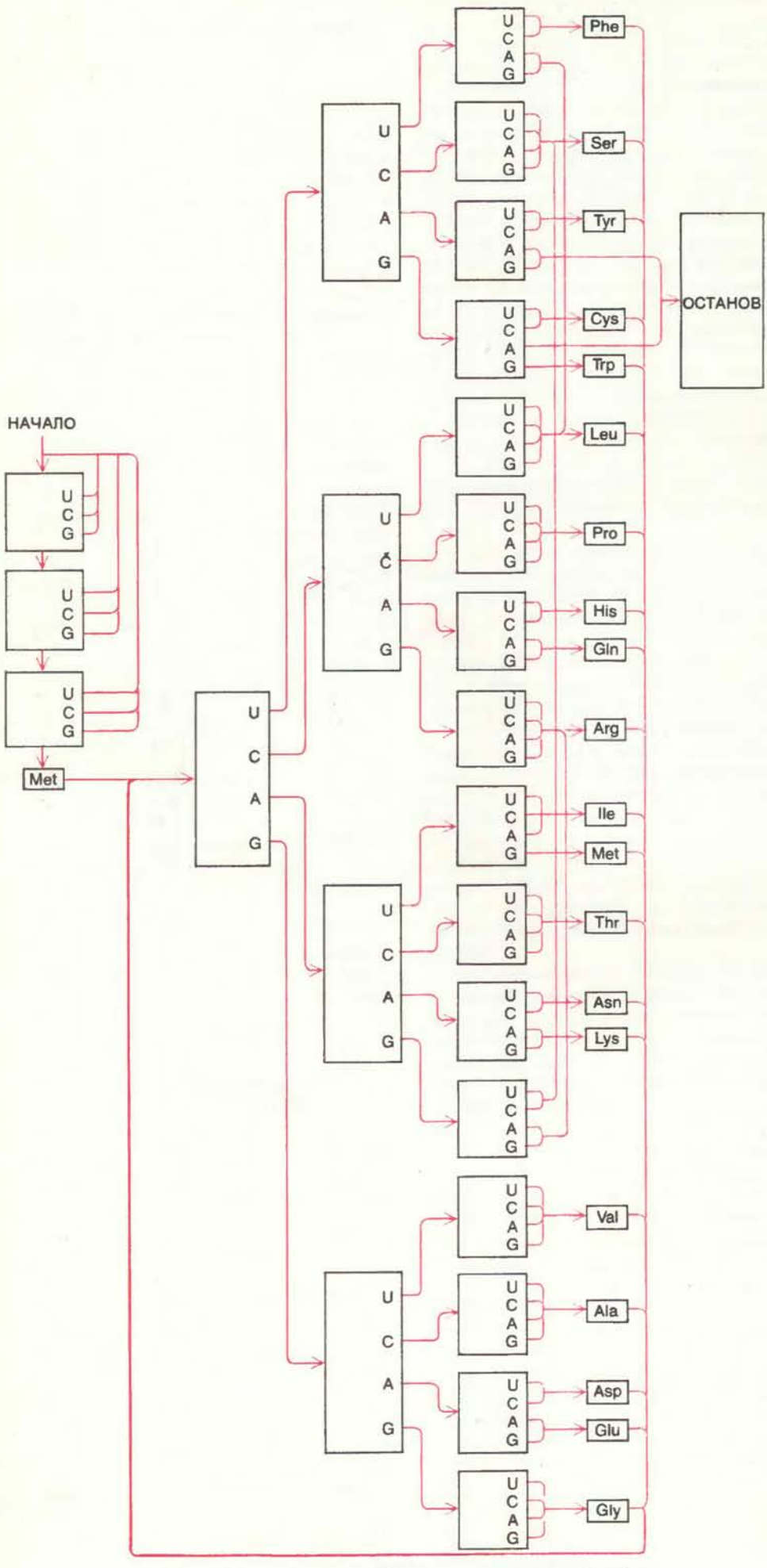
В состоянии D признаются только литеры C, L, X, V и I, имеющие значение, меньшее D. Та же самая последовательность допустима в состоянии C (поскольку C может повторяться), а в состоянии L признаются лишь литеры X, V и I. Правило, согласно которому происходят переходы, вам уже, наверное, понятно. Состояния упорядочены в иерархию, и, как только достигается данный уровень, автомат больше не может вернуться к более высокому уровню; на уровне «половинок» он даже не может остаться. К тому времени, как достигается состояние I, допускается лишь повторный символ I либо пробел. Пробел, поступивший в этот момент либо в любой другой момент после появления первой литеры, указывает на конец «жетона» и переводит автомат обратно в начальное состояние, когда он готов получить следующее число.

Ни один из известных мне языков программирования не допускает ввод чисел, записанных римскими цифрами, но фактически все такие языки имеют средства обращения с числами, записанными арабскими цифрами. Методы распознавания аналогичны, но имеется большее разнообразие форматов. Простые целые числа, например 137, могут обрабатываться в принципе автоматом с одним состоянием, но числа, состоящие из нескольких частей, такие, как $+ 6.625 \cdot 10^{-27}$, требуют более



Лексический анализатор для языка, состоящего из римских цифр

WWW.SCIAM-MAGAZINE.NAROD.RU



Конечный автомат расшифровывает генетический код

сложного лексического анализа.

Систему рибосом и тРНК можно рассматривать как лексический анализатор, который распознает биологически осмысленные последовательности нуклеотидов в молекуле мРНК. Для того чтобы последовательность была принята, она должна начинаться с инициирующего кодона и заканчиваться одним из трех терминирующих кодонов; между этими границами допускается любая комбинация входных символов U, A, G и C.

Лексический анализ — это первый шаг в процессе компиляции. После лексического анализатора вступают компоненты компилятора, носящие названия «программа грамматического разбора» и «генератор кода». В качестве входной последовательности программа грамматического разбора берет «жетоны», распознанные лексическим анализатором, и анализирует их синтаксические отношения; на этом этапе происходит более глубокое понимание компилятором смысла операторов программы, которые он транслирует. Программу, которая выполняет функции, определенные разобранными операторами, генератор кода пишет на объектном языке.

В рассмотренных здесь игрушечных языках задачи программы грамматического разбора и генератора кода тривиальны. Транслированная форма оператора на языке, использующем римские цифры, может оказаться просто эквивалентом этого числа, записанного арабскими цифрами. Ее можно было бы генерировать, следуя такой стратегии. Перед лексическим анализом «жетона» отводится ячейка памяти и туда засылается нуль. Затем каждый раз, когда лексический анализатор входит в состояние M, к содержимому ячейки прибавляется 1000; для состояния D прибавляется 500 и т.д. Когда лексический анализ закончен, в этой ячейке содержится число, записанное римскими цифрами. Заметим, что этот игрушечный компилятор уже не будет конечным автоматом, поскольку у него есть вспомогательная память.

Компилятор для генетического кода даже проще и полностью может быть построен в рамках конечного автомата. Транслируемая «программа» является последовательностью стандартных трехбуквенных символов для аминокислот; символы могут быть генерированы как выходная последовательность состояний лексического анализатора, распознающего кодоны. Три состояния, соответствующие трем терминирующим кодонам, не имеют выходной последовательности.

Создание настоящего компилятора для достаточно широкого языка, который имел бы практическую пользу, не слишком простое дело, но лежащая в основе его структура конечного авто-

WWW.SCIAM-MAGAZINE.NAROD.RU

Иерархия Хомского конечных и бесконечных автоматов

мата может по крайней мере помочь найти принцип организации. Если синтаксис языка определен достаточно точно, то часть работы даже можно автоматизировать: выполнить при помощи программы-компилятора нашего компилятора. Входной последовательностью для нее служит формальное описание языка, а выходной последовательностью — другая программа, которая транслирует операторы на этом языке. Насколько мне известно, еще не написан компилятор компилятора компилятора.

РАСПОЗНАВАНИЕ «жетонов» лексическим анализатором уже само по себе в некотором роде грамматический разбор, а множество всех возможных последовательностей символов «жетона» является своего рода языком. В сущности, это бесконечный язык: если только не наложено какое-нибудь искусственное ограничение на длину индивидуальной последовательности, можно образовать бесконечное множество распознаваемых «жетонов». Как может автомат, имеющий лишь конечное число частей, распознавать бесконечные последовательности правильно составленных операторов и отбрасывать бесконечное число неправильных? Разгадка заключена в самой структуре языка. Если операторы бесконечного языка должны распознаваться конечным автоматом, они должны быть образованы по строгим правилам.

Эти правила были сформулированы в 1956 г. С. Клини; они определяют класс языков, названных регулярными языками или регулярными множествами. Клини доказал, что конечный автомат может распознавать язык только тогда, когда этот язык регулярен, и, кроме того, что каждый регулярный язык может быть распознан некоторым конечным автоматом: в конце концов можно построить автомат, в котором имеется состояние для каждого возможного выражения языка. Далее, если язык бесконечен; должен быть возможен грамматический разбор всех операторов посредством прочитывания одного символа слева направо, либо начиная с конца (при этом не допускается ни забегания вперед ни возвращения назад). Если допустимость какого-либо символа обусловлена присутствием другого символа, то обусловливающий символ должен находиться непосредственно слева.

Второе правило является немедленным следствием ограничений конечного автомата, который не может ни предвидеть свои будущие состояния, ни сохранить память о прошлых состояниях; он должен выбрать переход на основе только текущего состояния и текущего входного символа. Именно по этой причине конечный автомат не может обрабатывать числа, образованные римскими цифрами при помощи вычитания. Если записано выражение XI и автомат интерпретирует его как 11, он не может вернуться назад и пересмотреть значение, если следующая литера оказывается V. Многие другие функции управляются с помощью такого ограничения. Например, невозможно построить конечный автомат, который читал бы последовательность двоичных разрядов и определял бы, равно ли число единиц числу нулей. Аналогично, хотя конечный автомат может складывать двоичные числа, он не может их перемножать; я предоставляю читателю догадаться, почему.

ЗА ПРЕДЕЛЫ конечных автоматов и регулярных языков простирается иерархия более мощных машин и более общих языков. Она называется иерархией Хомского, по имени лингвиста Н. Хомского, который исследовал различные формальные языки как возможные модели естественных языков. Наиболее общие языки были созданы при помощи смягчения ограничения на грамматические правила регулярных множеств; автоматы же были построены добавлением к исходной модели конечного автомата элементов памяти.

Следующий автомат в этом ряду называется автоматом с магазинной памятью. Он состоит из конечного автомата с добавлением массива памяти, которая имеет бесконечную емкость, но причудливую организацию. Эта память принимает форму стека, подобного стопке подносов в кафетерии. Для занесения единицы информации в память она должна быть помещена на вершину стека; восстановлена она будет не раньше, чем будут удалены все расположенные сверху единицы. Таким образом, последний из поступивших уходит первым.

Язык, распознаваемый автоматом с магазинной памятью, называется контекстно-свободным языком. При разборе его операторов допустимость символа может зависеть и от символа, стоящего непосредственно слева, и от символа, стоящего непосредственно справа. Эта двусторонняя зависимость разрешается, потому что каждый символ, допустимость которого нельзя немедленно установить, может быть сохранен в стеке, пока не будет разрешена неясность. Следовательно, автомат с магазинной памятью может оперировать с числами, записанными римскими цифрами и образованными при помощи вычитания, и он может распознавать выражения с одинаковым числом единиц и нулей (или иных символов, таких, как правые и левые скобки). С другой стороны, он не может обнаруживать предложения с равным числом символов трех видов, таких, например, как 0, 1, 2. Большинство языков программирования контекстно-свободные, а программа разбора компилятора обычно представляет собой автомат с магазинной памятью. Многие компьютеры имеют аппаратные средства для организации части памяти в виде стека. Язык программирования Forth за основную структуру памяти принимает структуру стека. Разумеется, стек в любой реальной машине не может иметь бесконечную глубину.

WWW.SCIAM-MAGAZINE.NAROD.RU

Контекстно-свободные языки оправдывают свое название, поскольку на распознавание любого символа могут прямо повлиять только его два ближайших соседа, а не более широкий контекст, в котором он находится. Снятие этого ограничения порождает контекстно-зависимые языки и еще больше увеличивает сложность интерпретации. Теперь далеко отстоящие друг от друга символы могут взаимодействовать; в самом худшем случае первый символ выражения нельзя интерпретировать, пока не будет прочитан последний. В обмен на дополнительную сложность получаем несколько бóльшие возможности. Автомат, основанный на контекстно-зависимых языках, может определять, содержится ли в выражении одинаковое число символов трех видов.

Автомат, который может распознавать контекстно-зависимый язык, является линейно-ограниченным автоматом. В дополнение к обычному устройству конечного автомата он имеет память, организованную так, что любая ячейка памяти доступна в любой момент; это автомат со случайным доступом. Емкость памяти конечна, но предполагается, что она достаточно велика, чтобы вместить любую входную последовательность, которая поступает в автомат. Линейно-ограниченные автоматы, по-видимому, дают хорошее приближение к модели цифровой вычислительной машины фон Неймана. Но, как ни странно, соответствующие контекстно-зависимые языки программирования, по-видимому, мало распространены; очевидно, более простая контекстно-свободная структура почти всегда позволяет добиться желаемой выразительности.

Все описанные выше языки имеют одно общее свойство: говорят, что они рекурсивны. Для нас это означает, что можно вообразить процедуру порождения всех возможных «способов выражения» в языке в порядке возрастания длины. Отсюда следует, что существует гарантированный метод установления того факта, принадлежит ли данному языку данный оператор конечной длины, — для этого просто надо породить все операторы вплоть до данной длины и сравнить их.

Имеются языки, которые не удовлетворяют даже этим простейшим стандартам обрабатываемости. Для них существует лишь один возможный распознающий автомат, «последнее средство», — машина Тьюринга, конечный автомат со свободным доступом в бесконечную память. В описании, данном Тьюрингом, память представлена в виде бесконечной в обоих направлениях ленты, разделенной на клетки, в которые конечный автомат может записывать, считывать с них или стирать записанное.

Теперь с высот машины Тьюринга четче вырисовывается взаимосвязь вычислительных устройств более низких уровней. Линейно-ограниченный автомат — это просто машина Тьюринга с конечной лентой. Автомат с магазинной памятью имеет ленту, не ограниченную в одном направлении, но «головка» для чтения и записи на ленте всегда остается фиксированной над последней непустой клеткой. Конечный автомат — это машина Тьюринга вовсе без ленты.

Читателям, падким на все новое и желающим заняться грамматическим разбором нерекурсивных языков, пора спешить приобрести машину Тьюринга. Следует только предупредить их, что этот чудо-компьютер имеет свои слабые стороны. Существуют языки с такими нелепыми грамматиками, что даже машина Тьюринга не сможет распознать их операторы за конечный отрезок времени. Пока такие языки нашли весьма ограниченное применение в мире вычислительных машин, но люди как-то ухитряются на них говорить.

Издательство **МИР** *предлагает:*

Эрвин Фрид

ЭЛЕМЕНТАРНОЕ ВВЕДЕНИЕ В АБСТРАКТНУЮ АЛГЕБРУ

Перевод с английского

Книга известного венгерского математика Эрвина Фрида содержит популярное, но достаточно строгое изложение основ абстрактной алгебры. В доступной, яркой и убедительной форме представлены теория групп и полугрупп, колец, тел и векторных пространств, рассмотрены структуры и операции над множествами, соотношения между структурами.

В книге отражены также основные направления развития современной алгебры. Многочисленные примеры и задачи призваны помочь читателю активно овладеть специфическими особенностями алгебраического мышления. Текст книги сопровождается выразительными иллюстрациями.

Книга рассчитана на студентов, инженеров, ученых смежных специальностей и всех, кто интересуется современной математикой.

1979, 120 страниц с иллюстрациями.
Цена 1 р. 50 к.

ИЗДАТЕЛЬСТВО МИР